CSE422: Artificial Intelligence

Car Price Prediction

Determining Market Prices of Cars using ML

Table of contents

1. Introduction…… 4

2. Motivation…….... 4

3. Dataset Description…… 5

4. Dataset Pre-Processing……..…… 8

4.1. Handelling Null Values

4.2. Imputing Values in Null cells

4.3. Using Interval

4.4. Encoding categorical features

4.5. Oversampling

5. Feature Scaling…….. 9

6. Dataset Splitting…… 9

7. Model Training…….. 10

8. Comparison analysis…… 14

9. Model Testing……15

9.1. K-Nearest Neighbor (KNN)

9.2. Support Vector Machine (SVM)

9.3. Logistic Regression

9.4. Decision Tree

10. Conclusion…… 15

11. Future Work…….. 16

Introduction

Cars are available in a wide range of selling price, features such as transmission, fuel type and also brand, manufacturing date, seller type & owners. A key component of consumer strategy is the estimate and forecast of prices based on those. An article by *Remington Hall* states forecasting allows for data-driven strategy development and well-informed corporate decision making, achievable via Artificial Intelligence, specifically Business intelligence or BI.

Hence, why our study focuses on training machine learning models using readily accessible information on the numerous features and price ranges of cars in the market to make the price prediction more accurately untestable for the customers.

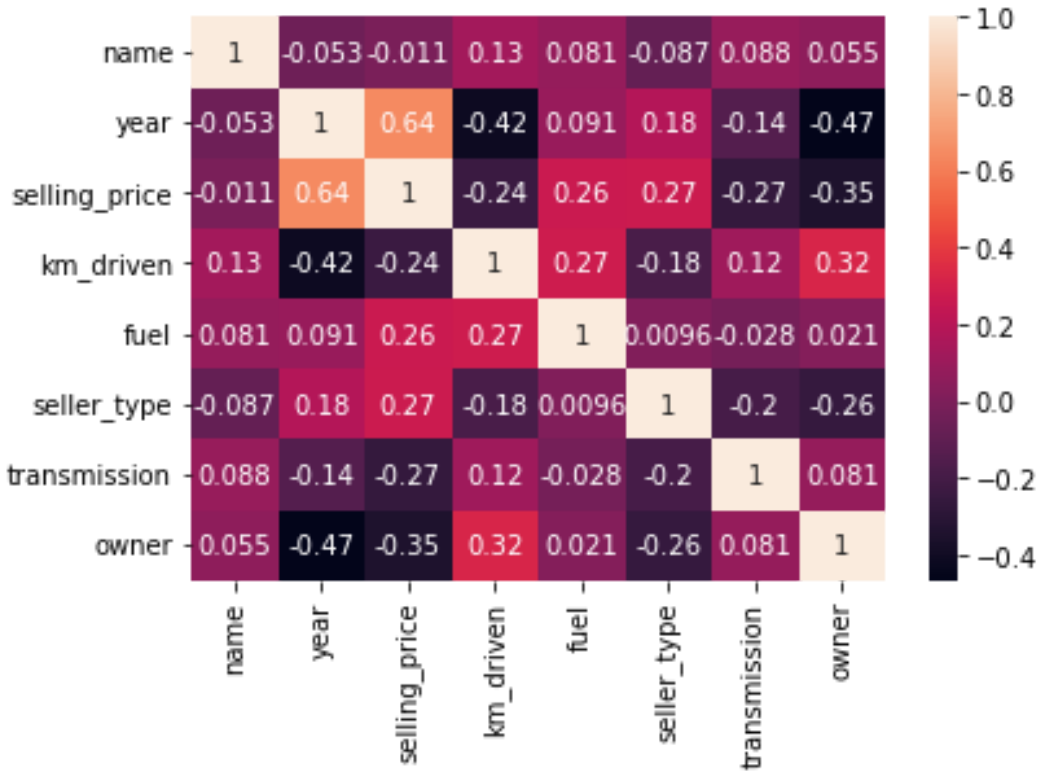
Motivation

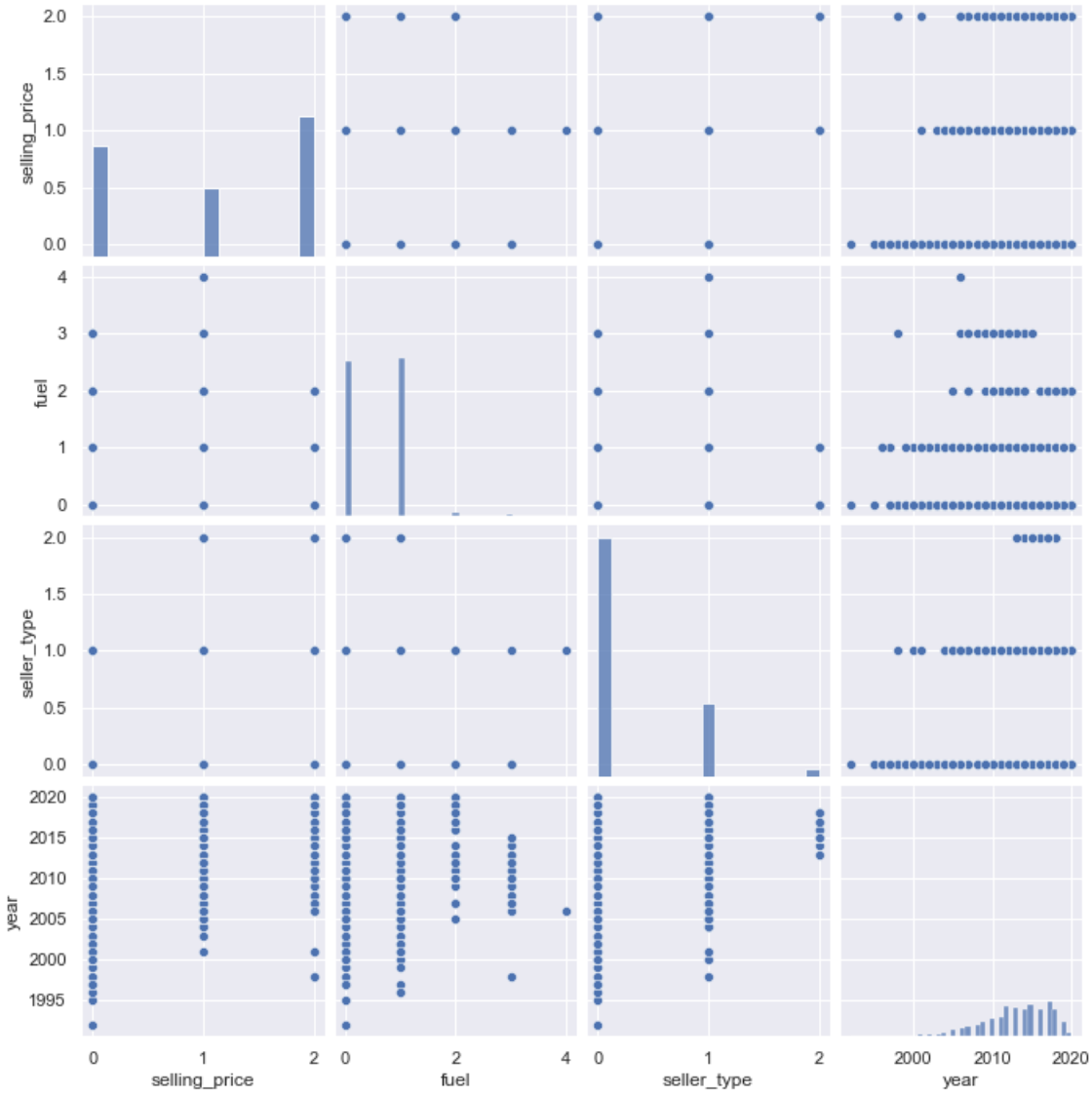
Before buying a car, we need to go through a lot of things. Through this machine learning on car price prediction, we can get a range(low, medium, high) on price of the desired car according to the needs and this is done based on the car type (transmission), km driven, model, manufacturing year, which is very convenient for the buyer to know the current demand and price of the car.

Dataset Description

In our project we used a dataset collected from Kaggle which can be found using this link: https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho. Our data sheet CAR DETAILS FROM CAR DEKHAO.csv is a Comma Separated Values file, a plain text file containing a list of data. It has been published it on the data-publishing site Kaggle under Car Price predection.

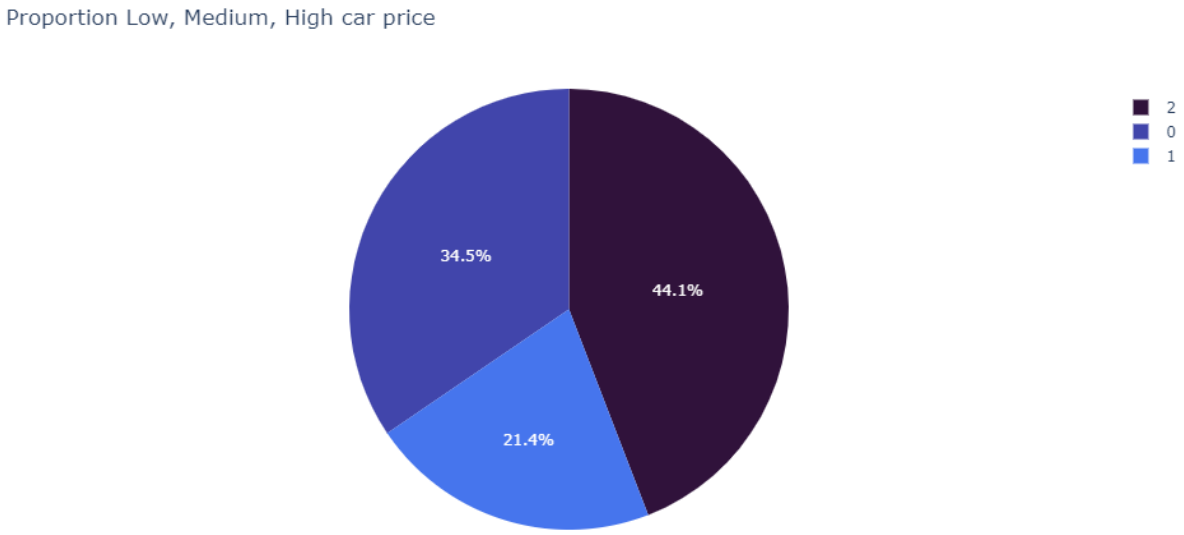
The set consists of 4340 rows and 8 columns including Name, Year, Selling price, Kilometer driven, Fuel, Transmission, Owner. There are 7 features and the Label ‘selling price’ is discrete numerical datatype. The features includes discrete, ordinal and nominal datatypes.



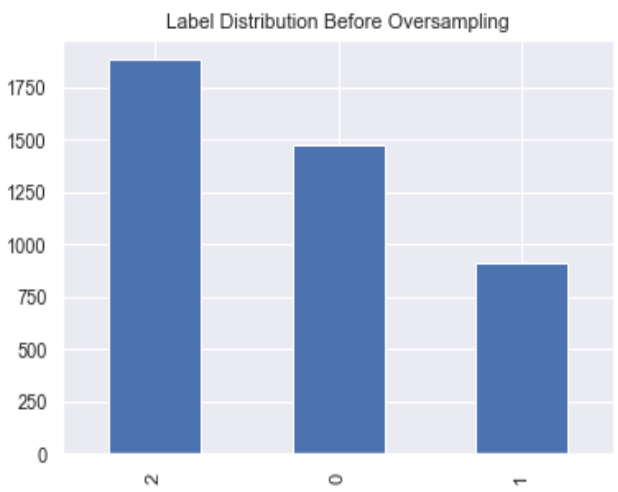


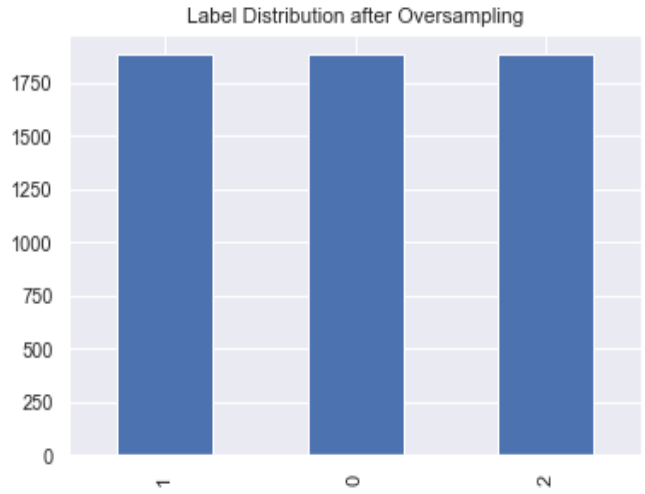
From the correlation matrix and scatterplot, we can see that our label ‘selling price’ has positive correlation with year, fuel and seller type and so we will choose these 3 features for our models.

Even though selling_price has 443 unique values we covert them into low,medium and high range where 34.5% values are low, 21.4% values are medium and 44.1% are high.



In our dataset skewness: -0.188553 which means most values are concentrated on the right of the mean, with extreme values to the left.





From the graph above, we can see the values in Label are not balanced. Thus, we use oversampling to balance our Label. After oversampling we get the graph below.

Dataset Pre-Processing

Handelling Null Values

Applying *.isna().sum()* on the imported dataset returns the amount of null values. There were missing values in all the columns. So, we dropped the rows having null values except for km_driven using .dropna.

Imputing Values in Null cells

For km_driven column we used SimpleImputer to impute mean of the column to replace null values.

Using Interval

As our Label has 443 unique values where min is 20000 and max is 8900000 so we are ranging the values from 20000-250000 to low, 250000-400000 to medium and 400000-9000000 to high.

Encoding categorical features

The features fuel and ower_type are in nominal datatype. To covert them into discrete values we used .map where we also could have used one hot encoding.

Oversampling

The label was not balanced so we used oversampling to make it unbiased.

Feature Scaling

Standardization is a scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. Furthermore, it can be helpful in cases where the data follows a Gaussian distribution. However, this is not necessarily true. Unlike normalization, standardization does not have a bounding range. So, even if there are outliers in data, they will not be affected by standardization.

We used both MinMaxScaler and Standard Scaler but both gave lower 67% accuracy where without scaling we got 68% accuracy using KNN. So we did not use scaling in the algorithms ahead.

Dataset Splitting

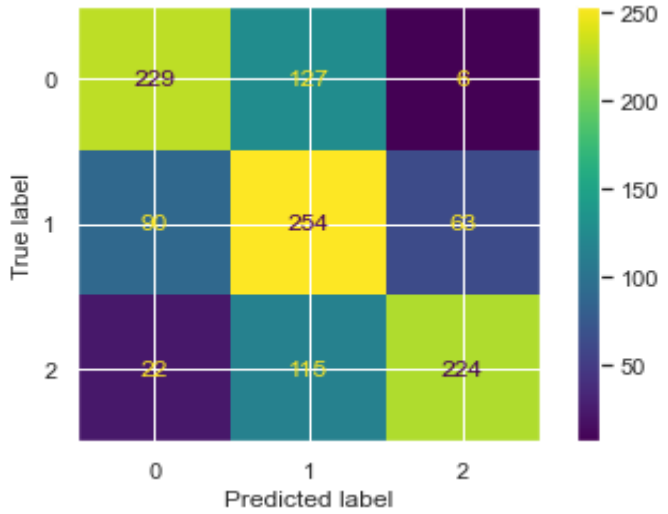
In our Y there was our label selling_price and in X we had year, fuel and seller_type. We splitted our data where training dataset had 80% and testing dataset 20% of the total data.

Model Training

K-Nearest Neighbor (KNN)

KNN calculates the Euclidean distance (diagonal distance) between the query point and the k-number of the nearest neighboring points. Then chooses the class label based on the highest frequency label. It is a supervised learning classifier that analyzes proximity to classify or anticipate how a single data point is grouped.

In our program, we import *KNeighborsClassifier* from *sklearn.neighbors*, a Scikit-Learn module, and model train with the default value of k being 3. This produces a poor accuracy score of 66.0% before training on the dataset and 60.0% after training on the dataset.

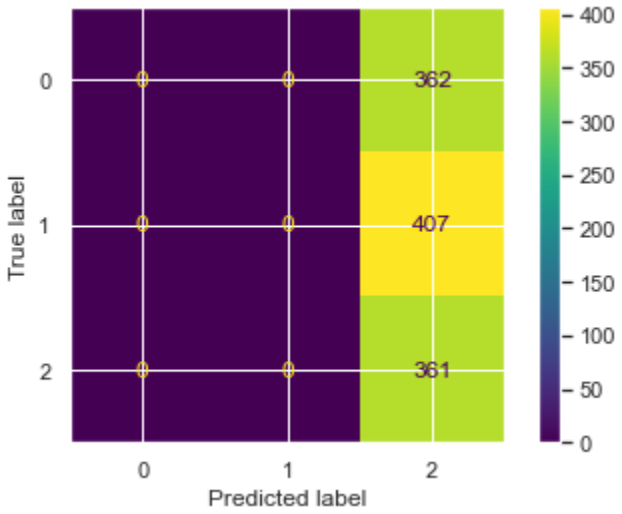


	precision	recall	f1-score	support
0	0.56	0.76	0.65	362
1	0.51	0.43	0.47	407
2	0.75	0.60	0.67	361
accuracy			0.59	1130
macro avg	0.61	0.60	0.59	1130
weighted avg	0.60	0.59	0.59	1130

Support Vector Machine (SVM)

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

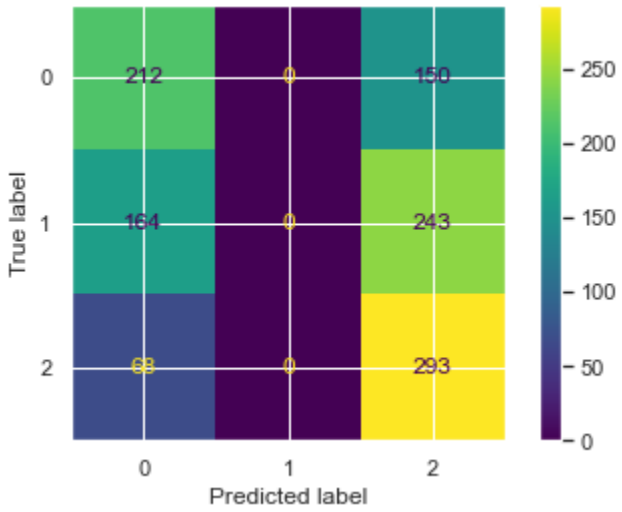
From *sklearn.linear_model*, we import *SVC* to train the model. Doing so gives an accuracy of 31% on the dataset, a score significantly lower than the KNN model.



Logistic Regression

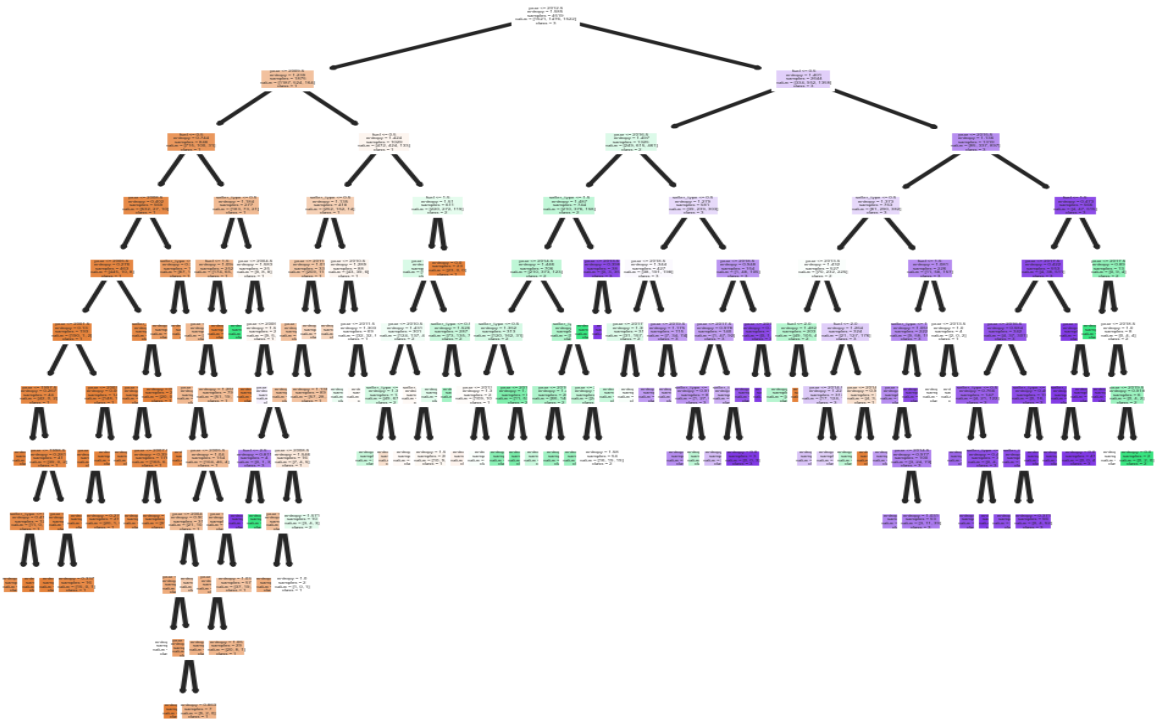
Logistic Regression, a supervised learning classifier, is a statistical model which predicts the probability of a binary event occurring on a given input dataset of independent variables. Therefore the output of the dependent variable is a probability between 0 and 1 (inclusive).

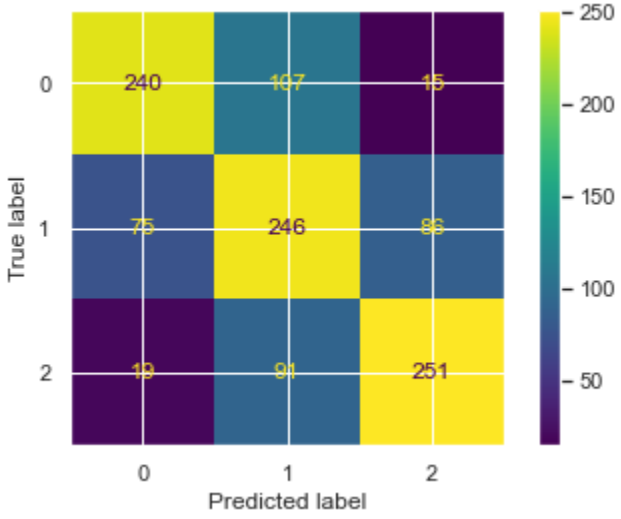
From *sklearn.linear_model*, we import *LogisticRegression* to train the model. Doing so gives an accuracy of 47% on the dataset, a score significantly lower than the KNN model.



Decision Tree

Classification and regression issues can be resolved using a decision tree, a supervised learning approach. It is a tree-structured classifier in which each leaf node represents the classification outcome, each internal node represents a feature and each branch is for decision-making.

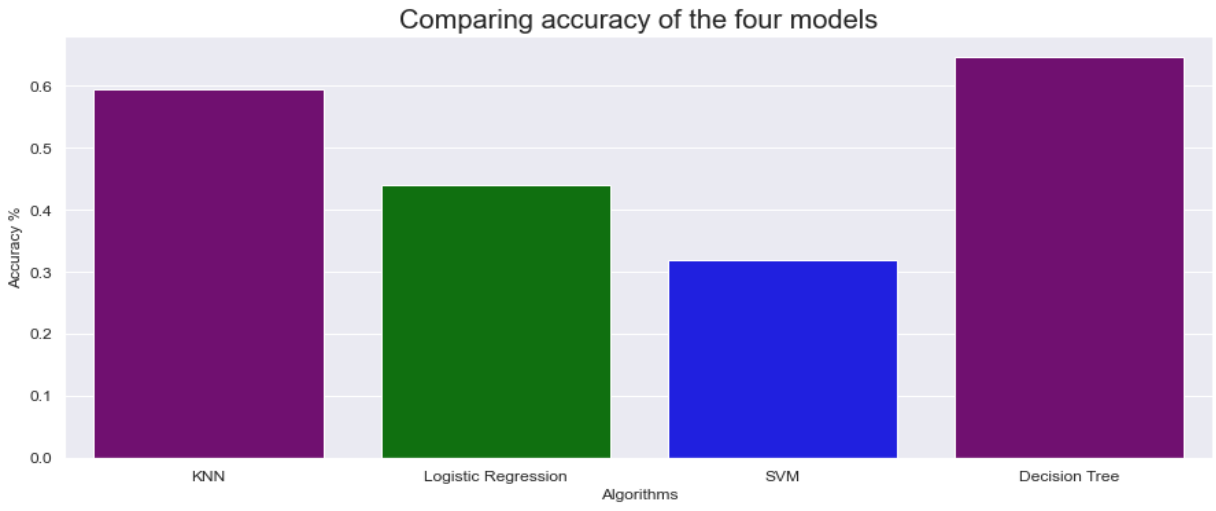




From *sklearn.tree*, we import *DecisionTreeClassifier* to train the model using the decision tree learning algorithm. This results in an accuracy score of 66.0% which is higher than the Naive Bayes model but not as high as the Logistic Regression model.

Comparison analysis

As per demonstrations of the four models, it is evident that the Decision Tree model produces the highest accuracy score (67.0%) before training on the dataset, whereas the SVM model produces the lowest accuracy score (31.0%). And lastly, the KNN model generates a score of 60.0%, which sits between the Decission Tree and the Logistic Regression models. In conclusion, the Decision Tree model can best predict the price of car prices relative to the given dataset. The visual representation of the final results are shown below through a bar chart which gives a better overview of the outcomes.



Model testing

In this part, we tested our modeling using unseen values. We gave Name:Maruti Celerio Green VXI year:2017, km_driven:78000 ,fuel:CNG, seller_type:Individual, transmission:Manual, Owner:First where the selling_price is 365000 so our model should predict selling price: medium. Thus, we used KNN algorithm and our model predicted correctly.

Conclusion:

We can conclude that our machine model can predict a car price quite accurately depending on the given data. Therefore, taking decisions for newer vehicle and car products can both be aided by this learning model.

Future work:

We want to increase the accuracy of our machine learning algorithm in future and also we want to work on some other CNN models that are not done here.

Reference:

1. Hall, R. (2020). *Why Forecasting is Important for Business Success*. Baass.com. https://www.baass.com/blog/why-forecasting-is-important-for-business-success

2. Nehal Birla. (2020). *Vehicle dataset*. Kaggle.com.

   https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho